In The United States Patent and Trademark Office

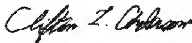| For:			
Computer Cluster With Second-Node Instance of Application Having Access to State Snapshot of First-Node Instance of Application			
Applicant:		Attorney Docket No.:	
Kenneth PRIDDY et al.		200314321-1	
Serial No.:	Filed:	Art Unit:	Examiner:
10/806,261 (1933)	2003-Mar-31	2192	Amine RIAD

Director For Patents
PO Box 1450
Alexandria, VA 22313-1450

Appeal Brief

(Identification Page)

This is an Appeal to the Board of Patent Appeals and Interferences from the Final Action mailed 2007-Oct-16. A Notice of Appeal was submitted on 2008-Jan-16.

Respectfully submitted
Kenneth PRIDDY et al.
by

Clifton L. Anderson
Reg. No. 30,989
(408) 257-6070

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real parties in interest are

Hewlett-Packard Company, a Delaware corporation; and

Hewlett-Packard Development Company, L.P., a Texas limited partnership and wholly owned affiliate of Hewlett-Packard Company, and assignee of record.

RELATED APPEALS AND INTERFERENCES

None.
There are no related appeals or interferences.

STATUS OF CLAIMS

Claims 1-12 are pending in the application.

Claims 1-12 are rejected.

The rejections of Claims 1-12 are on appeal.

STATUS OF AMENDMENTS

All amendments submitted have been entered.

There are no unentered amendments.

SUMMARY OF CLAIMED SUBJECT MATTER

When a computer of a high-availability cluster fails, another computer
can assume its mission, but must recalculate unsaved data that was in
volatile memory. Time lost to recalculation can be reduced
prospectively by frequently saving the state of volatile memory, but
this may require freezing an application program while a snapshot of
its state is saved to non-volatile disk-based memory. The present
invention reduces downtime for the application program by
transferring the snapshot first to volatile memory; then, with the
application continuing its mission, the snapshot in volatile memory is
transferred to disk. A refinement of the invention further reduces
application downtime by forming the snapshot in volatile memory in a
piecemeal fashion while the application continues its mission and
while some pieces of the snapshot are transferred to disk.

Independent Claim 1

Claim 1 relates to a computer cluster (Fig. 1, AP1; paragraph 17,
page 5, lines 7-16) comprising:

storage media (MD1);

a first computer (CP1) having a first instance (AA1; paragraph 19,
page 5, lines 23-24)) of an application program installed, said
application program having instructions, said first computer including,

volatile memory (RM1);

processing means (DP1 and DT1)

for executing instructions of said first instance of said
application program so as to modify data (AD) stored in
said volatile memory,

for creating a snapshot (SD) of said data while said first instance of said application program is running, said snapshot being stored in said volatile memory, and

for, while said first instance of said application continues to modify said data so that it diverges from said snapshot, transferring said snapshot from said volatile memory to said storage media, and

a second computer (CP2) having a second instance (AA2) of said application program installed (paragraph 19, page 6, lines 1-4), said second computer including means for accessing (DI2; paragraph 18, page 5, lines 17-18) said storage media so that said second instance of said application can access said snapshot as stored on said storage media (Fig. 2, steps S3B and S5A, page 8, lines 6-15).

Claim 2

Claim 2 depends from Claim 1 and further requires that the recited processing means includes

a data processor (Fig. 1, DP1; paragraph 17, page 5, lines 7-14)

for executing instructions of said first instance (AA1) of said application program so as to modify data (AD; paragraph 21, page 6, lines 18-19) stored in said memory, and

for creating said snapshot (SD; paragraph 20, page 6, lines 9-11) of said data while said first instance of said application program is running, said snapshot being stored in said volatile memory (paragraph 20, page 6, lines 11-13), and

a transfer processor (DT1) for transferring said snapshot from said volatile memory to said storage media while said first instance of said application program is running (page 6, lines 13-18).

Independent Claim 7

Claim 7 relates to a method (Fig. 2; M1; paragraphs 23-26, page 7, line 13 to page 9, line 7) comprising:

executing (S1; paragraph 23, page 7, lines 14-16) a first instance (Fig. 1, AA1) of an application program on a first computer (CP1) of a computer cluster (AP1) so as to generate a series of memory states (AD);

creating (Fig. 2, S2; paragraph 23, page 7, lines 16-18) a snapshot (SD) of one of said states; and

at least partially during a state (including ADM) that differs from the state represented in said snapshot, transferring (Fig. 2, S3A; paragraph 24, page 7, lines 27-29) said snapshot to storage media (Fig. 1, MD1) accessible by a second computer (CP2) of said computer cluster.

Claim 8

Claim 8 depends from Claim 7 and further requires executing (S3B, S5A; paragraph 24, page 8, lines 6-10; paragraph 25, page 8, lines 11-15) a second instance of said application program on a second computer of said computer cluster using said snapshot as a starting state.

Claim 10

Claim 10 depends from Claim 8 and further requires that said executing (S3B; paragraph 24, page 8, lines 6-10) a second instance follows said transferring without an intervening detection of a failure (paragraph 26, lines 23-26).

Claim 11

Claim 11 depends from Claim 7 and further requires that said transferring is effected by a data transfer processor (Fig. 1, DT1; paragraphs 20-21, page 6, lines 12-16), not used in executing said first instance of said application.

Claim 12

Claim 12 depends from Claim 7 and further requires that executing is effected by a data processor (Fig. 1, DP1) that stores processor state data (SNP) internally, said snapshot (SD) including said processor state data (paragraph 9, page 3, lines 11-12; paragraph 22, page 7, lines 1-11).

GROUNDS OF REJECTION TO BE REVIEWED

All outstanding grounds of rejections are to be reviewed. These grounds are set forth below.

The rejections of Claims 1-12 for anticipation by U.S. Patent No. 7,058,846 to Kelkar *et al.*, "Kelkar" herein, are to be reviewed.

ARGUMENTS

[01] **ARGUMENTS FOR REVERSING REJECTIONS FOR
ANTICIPATION BY KELKAR**

[02] For the purposes of these grounds of rejection, the claims are
divided into seven groups. Group 1 includes Claims 1 and 3-5;
Group 2 includes Claims 2 and 6; Group 3 includes Claim 7; Group 4
includes Claims 8 and 9; Group 5 includes Claim 10; Group 6 includes
Claim 11; and Group 7 includes Claim 12.

[03] **Group 1**

[04] "creating a snapshot"

[05] Claim 1 refers to a first instance of an application program that
modifies data stored in volatile memory. Herein, the claimed data is
referred to as "application data" to distinguish it from other data, e.g.,
data not modified by the application program). Claim 1 requires
creating a snapshot of said application data. Kelkar does not disclose
creating a snapshot of application data. For this first reason, the
rejection of Claim 1 for anticipation by Kelkar should be reversed.

[06] As the Final Action points out, Kelkar does disclose an
application program.

> In an example embodiment, nodes 110A and 110B are
> configured as servers for the same application program.
> *(column 5, lines 8-9)*

> The RAM is generally the main memory into which the
> operating system and application programs are loaded and
> typically affords at least 66 megabytes of memory space.
> *(column 10, lines 11-14)*

[07]   Kelkar does not mention or describe the application program
beyond what is presented in these two passages. The application
program is not given a reference number and is not represented in the
figures. While Appellants will stipulate that the disclosed application
program inherently manipulates data in volatile memory, Kelkar does
not actually disclose this application data, let alone what happens to it.
In particular, Kelkar does not disclose creating a snapshot of
application data.

[08]   The Final Action makes several contrary assertions. First, the
Final Action asserts that the first passage above discloses modification
of application data in volatile memory, but there is no mention of data
in that passage. Second, the Final Action asserts that the application is
represented in Figure 2 as "item" 110A. However, Kelkar refers to a
"node" 110A, which is a computer or server on which the application is
run. While various instances of system software are represented
explicitly in Fig. 2 and elsewhere in Kelkar, the application program is
not explicitly represented. Third, the Final Action asserts that the
claimed snapshot is disclosed at Kelkar, column 3, lines 40-44, which
passage appears in the following full paragraph.

The present invention provides a method, system, and
computer program product to make resource configuration
information available to nodes in a cluster in as close to real-
time as possible with minimal overhead. Resource
configuration data are synchronized at nodes in the cluster,
thereby enabling a node having synchronized configuration
data to resume operations of a failed node. These
operations include storage management services that allow
configuration changes to be made dynamically to storage
resources. Examples of resource configuration changes
include adding a new disk to a storage array, creating a
snapshot of a storage area to back up data at given point in
time, and so on. *(Kelkar, column 3, lines 32-44)*

[09]   This paragraph contains the only mention of a "snapshot" in
Kelkar. However, the subject of the snapshot is a "storage area" rather
than application data in volatile memory. The reference to "adding a
new disk to a storage array" suggests that the snapshot is of data
stored on an array of hard disks. In any event, the Final Action does
not establish the contrary, i.e., that Kelkar discloses creating a
snapshot of application data stored in volatile memory. **Since the
Final Action fails to establish that Kelkar discloses the Claim 1
limitation of creating a snapshot of application data in volatile
memory, the rejection for anticipation of Claim 1 by Kelkar should
be reversed.**

[10] Storing the snapshot in volatile memory

[11] Claim 1 requires that the snapshot be stored in volatile memory.
Kelkar does not disclose this limitation. The only mention of a
snapshot by Kelkar, occurs in the column 3, lines 32-44 passage
quoted above. This passage does not mention where the snapshot is
stored. Those skilled in the art would consider it likely that the
snapshot is stored on hard disk, optical disk, or tape archive. Volatile
memory would be a poor choice for storing the snapshot because it
can be expensive and the snapshot can be lost in the event of a failure.
In any event, the Final Action fails to establish that Kelkar discloses the
Claim 1 limitation of storing a snapshot in volatile memory. **Since the
Final Action fails to establish that Kelkar discloses the Claim 1
limitation of storing a snapshot in volatile memory, the rejection for
anticipation of Claim 1 by Kelkar should be reversed for this second
reason.**

[12]   Transferring snapshot from volatile memory to storage media

[13]   Claim 1 requires a snapshot to be transferred from volatile
memory to storage media. Kelkar does not disclose this limitation.

[14]   The Final Action purports to find this limitation disclosed at
Kelkar, column 3, lines 40-40. (Appellant reads this as lines 40-41),
which passage refers to allowing "configuration changes to be made
dynamically to storage resources".   (This means that the storage
resources can be reconfigured without interrupting the application
program.) This passage says nothing about transferring a snapshot
from volatile memory to storage media. Thus, the Final Action fails to
establish that Kelkar discloses transferring a snapshot from volatile
memory to storage media. **Since the Final Action fails to establish
that Kelkar discloses the Claim 1 limitation of transferring a
snapshot from volatile memory to storage media, the rejection for
anticipation of Claim 1 by Kelkar should be reversed for this third
reason.**

[15]   While said application continues to modify said data

[16]   Claim 1 requires that the snapshot be transferred while the
application program continues to modify its data. Kelkar does not
disclose this limitation, as Kelkar does not mention application data or
indicate what happens to it. It is not inherent that an application will
continue to modify its data while a snapshot is being transferred to
storage media. Thus, the Final Action fails to establish that Kelkar
discloses transferring a snapshot while an application program
continues to modify data. **Since the Final Action fails to establish
that Kelkar discloses the Claim 1 limitation of transferring a
snapshot while the application program continues to modify its
data, the rejection for anticipation of Claims 1 and 3-5 (which
depend from Claim 1) by Kelkar should be reversed for this fourth
reason.**

[17]   **Group 2**

[18]   Claim 2 depends from Claim 1. **Thus, the four reasons given
above for reversing the rejection of Claim 1 apply as well to Claim 2.**

[19]   Claim 2 further requires a transfer processor as well as a data
processor. Kelkar does not disclose a separate transfer processor.

[20]   The Final Action asserts that Kelkar discloses a data processor at
Kelkar, Fig. 7, item 714 and a separate transfer processor at Kelkar,
Fig. 1, 102a. However, there are two problems with this attempt to
read Claim 2 on Kelkar: 1) Kelkar Fig. 1 and Kelkar Fig. 7 relate to
different systems, so item 102a and 714 are not shown in the claimed
combination; and 2) storage access interface 102a is not a processor.

[21]   Kelkar, Fig. 1 depicts a "typical system for storage resource configuration". This typical system is described to explain the problem that Kelkar's invention is intended to address. Fig. 1 does not depict an embodiment of Kelkar's invention, whereas Fig. 7 depicts an embodiment of Kelkar's invention. Obviously, storage access interface 102a and processor 714 are not present in combination in a single system. **Since the Final Action has failed to establish that Kelkar discloses the claimed combination of a data processor and a transfer processor, the rejection for anticipation of Claim 2 should be reversed for this fifth reason.**

[22]   Furthermore, item 102A does not qualify as a "transfer processor". The present specification states the following.

> Computer CP1 includes a data processor DP1, a transfer processor DT1 (which can be identical to data processor DP1 but assigned to a different task herein), volatile random-access memory RM1, a root hard-disk RD1, a disk interface DI1, and a network interface NI1. *(Present specification, paragraph 17, page 5, lines 9-13)*

[23]   This passage lists hardware components of computer CP1; transfer processor is a hardware component. On the other hand, Kelkar makes it clear that storage access interface 102A is software.

Storage access interface 102A represents a vendor-supplied interface for managing a particular storage resource. Storage access interface 102A is typically installed on a host computer system coupled to the storage resource, in this case, node 110A. Storage access interface 102A may be, for example, a dynamically linked library including functions that can be called to perform storage operations. Storage access interface 102A may include an application programming interface, a command line interface and/or a graphical user interface for managing storage resource 140. *(Kelkar, column 4, lines 8-17)*

[24] Thus, the Final Action fails to establish that Kelkar discloses the claimed transfer processor. **Since the Final Action has failed to establish that Kelkar discloses the claimed transfer processor, the rejection for anticipation of Claim 2 and Claim 6 (which depends from Claim 2) should be reversed for this sixth reason.**

[25] **Group 3**

[26] Creating a snapshot of a memory state

[27] Method Claim 7 requires creating a snapshot of a memory state. Kelkar does not disclose this limitation.

[28] Kelkar discloses creating a snapshot of a storage area. However, Kelkar does not disclose that the storage area is a memory state. Accordingly, the Final Action has failed to establish that Kelkar discloses creating a snapshot of a memory state. **Since the Final Action has failed to establish that Kelkar discloses the Claim 7**

limitation of creating a snapshot of a memory state, the rejection for anticipation of Claim 7 should be reversed.

[29] Transferring a snapshot

[30] Claim 7 requires transferring a snapshot. Kelkar does not disclose transfer of a snapshot. Note that if a snapshot is created in its eventual destination, there is no need to transfer it. Kelkar discloses creating the snapshot, but not transferring it. Thus, the Final Action has failed to establish that Kelkar discloses the limitation added by Claim 7. **Since the Final Action has failed to establish that Kelkar discloses the Claim 7 limitation of transferring a snapshot, the rejection for anticipation of Claim 7 should be reversed for this second reason.**

[31] **Group 4**

[32] Claim 8 depends from Claim 7. **Thus, the two reasons given above for reversing the rejection of Claim 7 apply as well to Claim 8.**

[33] Claim 8 further requires a second instance of an application program using a snapshot as a starting state. Kelkar does not disclose this limitation.

[34] The Final Action asserts this limitation is met by Kelkar's abstract.

> If a node that has made a resource configuration change
> fails, the resource configuration change is available for use
> by other nodes in the set, each of which can resume
> operations of the failed node. *(Kelkar, abstract, last sentence)*

[35] This sentence is to the effect that a second computer has up-to-date information on storage resources. This statement says nothing about the snapshot and nothing about a starting state for an instance of an application program. Thus, the Final Action has failed to establish that Kelkar discloses the limitation added by Claim 8. **Since the Final Action has failed to establish that Kelkar discloses the Claim 8 limitation of using a snapshot as a starting state of a second instance of an application program, the rejection for anticipation of Claim 8 and Claim 9 (which depends from Claim 8) should be reversed for this third reason.**

[36] **Group 5**

[37] Claim 10 depends from Claim 8. **Thus, the three reasons given above for reversing the rejection of Claim 8 apply as well to Claim 10.**

[38] Claim 10 further requires that a second instance of the application program be executed after the transfer of the snapshot without an intervening detection of a failure. Kelkar, does not disclose this limitation.

[39] The Final Action asserts that this limitation is disclosed in the passage quoted immediately below.

> To make resource configuration available to another node that can resume operation of node 110A upon failure, the invention synchronizes resource configuration data on multiple nodes in a clustering environment. *(Kelkar, column 4, lines 53-56)*

[40]   This passage says nothing about a second instance of an application program. Kelkar does disclose a computer cluster that permits a second instance of an application program to assume the state of a first instance of the application program upon detection of a failure of the computer on which the first instance was running. Kelkar does not disclose activation of the second instance without a failure detection. Thus, the Final Action has failed to establish that Kelkar discloses the limitation added by Claim 8. **Since the Final Action has failed to establish that Kelkar discloses the Claim 10 limitation of executing a second instance of an application program without detection of an intervening failure, the rejection for anticipation of Claim 7 should be reversed for this fourth reason.**

[41]   Group 6

[42]   Claim 11 depends from Claim 7. **Thus, the two reasons given above for reversing the rejection of Claim 7 apply as well to Claim 11.**

[43]   Claim 11 further requires "said transferring is effected by a data transfer processor not used in executing said first instance of said application." This limitation is analogous to that of Claim 2, which was not disclosed by Kevlar as established in the arguments for reversal of that claim. **Since the Final Action has failed to establish that Kelkar discloses the Claim 11 limitation of transferring by a processor not used for executing an application", the rejection for anticipation of Claim 11 should be reversed for this third reason.**

[44]   Group 7

[45]   Claim 12 depends from Claim 7. **Thus, the two reasons given above for reversing the rejection of Claim 7 apply as well to Claim 12.**

[46]   Claim 12 further requires that the snapshot contain state data from the processor that executes the first instance of an application program. Kelkar does not disclose this limitation.

[47]   The Final Action asserts that this limitation is disclosed at Kelkar, column 3, line 44, which is part of the "snapshot" paragraph quoted in the arguments regarding Claim 1. Line 44 reads "area to back up date at given point in time, and so on." This line cannot be interpreted as a disclosure that a snapshot contains processor state data. Thus, the Final Action has failed to establish that Kelkar discloses the limitation added by Claim 12. **Since the Final Action has failed to establish that Kelkar discloses the Claim 12 limitation of including processor state data in a snapshot, the rejection for anticipation of Claim 12 should be reversed for this third reason.**

Claims Appendix

1   1. *(previously presented)* A computer cluster comprising:

2   storage media;

3   a first computer having a first instance of an application program

4   installed, said application program having instructions, said first

5   computer including,

6   volatile memory;

7   processing means

8   for executing instructions of said first instance of said

9   application program so as to modify data stored in said

10   volatile memory,

11   for creating a snapshot of said data while said first

12   instance of said application program is running, said

13   snapshot being stored in said volatile memory, and

14   for, while said first instance of said application

15   continues to modify said data so that it diverges from said

16   snapshot, transferring said snapshot from said volatile

17   memory to said storage media, and

18   a second computer having a second instance of said application

19   program installed, said second computer including means for

20   accessing said storage media so that said second instance of said

21   application can access said snapshot as stored on said storage media.

1   2. *(previously presented)* A computer cluster as recited in Claim 1

2   wherein said processing means includes

3   a data processor

4   for executing instructions of said first instance of said

5   application program so as to modify data stored in said memory,

6   and

7   for creating said snapshot of said data while said first

8   instance of said application program is running, said snapshot

9   being stored in said volatile memory, and

10   a transfer processor for transferring said snapshot from said

11   volatile memory to said storage media while said first instance of said

12   application program is running.

1   3. *(original)* A computer cluster as recited in Claim 1 further

2   comprising a first cluster daemon running on said first computer for

3   causing said snapshot to be created.

1 4. *(previously presented)* A computer cluster as recited in Claim 1
2 further comprising a second cluster daemon running on said second
3 computer, said second cluster daemon providing:
4   for detecting a failure that prevents said first instance of said
5   application program from running on said first computer, and
6   for causing, in response to said detecting a failure, said
7   second computer to process said snapshot in accordance with
8   instructions of said second instance of said application program.

1 5. *(original)* A computer cluster as recited in Claim 1 wherein said
2 processing means provides for, in response to a write access of a
3 section of said volatile memory in accordance with instructions of said
4 first instance of said application program, copying data in that section
5 so that one instance of said data originally in that section is modified
6 and the other copy of data originally in that section is not modified.

1 6. *(original)* A computer cluster as recited in Claim 2 wherein said
2 data processing means maintains state data, said snapshot data
3 including at least some of said state data.

1   7. *(previously presented)* A method comprising:

2   executing a first instance of an application program on a first

3   computer of a computer cluster so as to generate a series of memory

4   states;

5   creating a snapshot of one of said states; and

6   at least partially during a state that differs from the state

7   represented in said snapshot, transferring said snapshot to storage

8   media accessible by a second computer of said computer cluster.

1   8. *(original)* A method as recited in Claim 7 further comprising

2   executing a second instance of said application program on a second

3   computer of said computer cluster using said snapshot as a starting

4   state.

1   9. *(original)* A method as recited in Claim 8 further comprising

2   detecting a failure that prevents execution of said first instance of said

3   application program, said detecting occurring after said transferring

4   and before said executing a second instance.

1   10. *(original)* A method as recited in Claim 8 wherein said executing a

2   second instance follows said transferring without an intervening

3   detection of a failure.

1   11. *(original)* A method as recited in Claim 7 wherein said transferring

2   is effected by a data transfer processor not used in executing said first

3   instance of said application.

1   12. *(original)* A method as recited in Claim 7 wherein said

2   executing is effected by a data processor that stores processor state

3   data internally, said snapshot including said processor state data.

EVIDENCE APPENDIX

None. No evidence is submitted with this Appeal Brief.

RELATED PROCEEDINGS APPENDIX

None. There are no related proceedings.